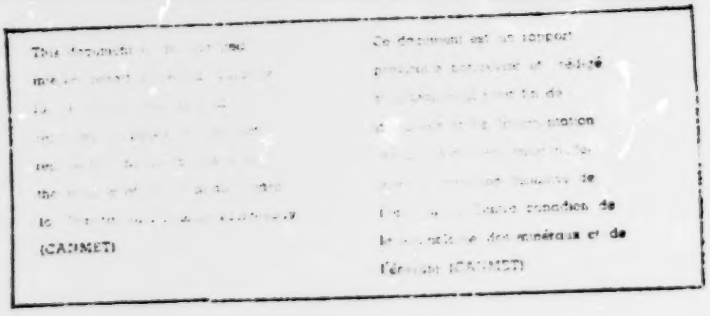
Energy, Mines and Resources Canada | Énergie, Mines et Ressources Canada

CANMET
Canada Centre for Mineral and Energy Technology | Centre canadien de la technologie des minéraux et de l'énergie

DIGITIZE/PLOTTER PROGRAM SOFTWARE

V.Srajer, N.J.Stuart, C.Neumiller, M.Heximer, W.Veldhuis

March 1987



24 pp

MICROMEDIA

ENERGY RESEARCH PROGRAM
COAL RESEARCH LABORATORIES
DIVISION REPORT ERP/CRL 87-20 (TR)

87-05150/

DIGITIZE/PLOTTER PROGRAM SOFTWARE

by

V. Srajer*, N.J. Stuart**, C. Neumiller***, M. Heximer***, W. Veldhuis***

ABSTRACT

In this report, the programs DIGITIZE and PLOTTER are presented and their functions are explained. The programs were written to help Surface Mining Laboratory staff to use the Calcomp digitizer and plotter. The programs' functions are to take an image on a piece of paper, record it in a data file, read the data from the data file, then plot it using the Calcomp plotter.

---

* Head, Operations Research, CANMET, CRL, Surface Mining Lab., Devon, Alb.
** Research Scientist, CANMET, CRL, Surface Mining Lab., Devon, Alberta.
*** Co-Op Students, University of Alberta, Edmonton, Alberta.

PROGRAMMES "DIGITIZE/PLOTTER"

par

V. Strajet*, N.J. Stuart**, C. Neumiller***, M. Heximer***, W. Velhuis***

RÉSUMÉ

Le présent rapport décrit les programmes DIGITIZE et PLOTTER ainsi que leurs fonctions. Les programmes ont été écrits pour aider le personnel du Laboratoire d'exploitation en surface à utiliser le numériseur et la table traçante Calcomp. Les fonctions des programmes sont les suivantes: prendre une image sur une feuille de papier, l'enregistrer dans un fichier de données, lire les données du fichier, puis tracer l'image à l'aide de la table traçante Calcomp.

*Directeur, Recherche opérationnelle, CANMET, LRC, Lab. d'exploitation en surface, Devon (Alberta).
**Chercheur, CANMET, LRC, Lab. d'exploitation en surface, Devon (Alberta).
***Étudiants inscrits à un programme de coopération, Université de l'Alberta, Edmonton (Alberta).

CONTENT

Page

ABSTRACT..... i
RESUME..... iii
INTRODUCTION..... 1
PROGRAM DIGITIZE..... 1
    How to Use Program..... 1
    A Sample Run 2
    Program Listing..... 2
PROGRAM PLOTTER..... 7
    Execution of Program..... 7
    How to Set up the Plotter..... 7
    Program Listing..... 7
CONCLUSION..... 20

INTRODUCTION

The programs written in VAX FORTRAN 77, are called DIGITIZE and PLOTTER. The former program's function is to reproduce an image on a piece of paper into a data file. The data file will then contain enough information to be able to reproduce the image on a plotter using the PLOTTER program.

The DIGITIZE program uses the digitizer's keypad for most of the data inputs. The text function requires the user to input the text, text size, and text angle through the terminal keyboard. The set up function of the Calcomp plotter is also introduced in this report.

PROGRAM DIGITIZE

How to Use Program.

The program is executed by the command:

RUN DIGITIZE <CR>

in the [IAN.FALL86.GRAPHS] directory. The screen will then display a command menu for the digitizer. The first few commands define the boundaries of the image. The main command asks for a 'LINE TYPE'. The line types are given as follows:

0 ... a change of colors (black, blue, red, green)
1 ... a set of points to be connected by lines
2 ... a vertical line containing two points
3 ... a horizontal line containing two points
4 ... a set of points to be connected by smooth curve
5 ... a line of text
6 ... a rectangle
... quit
D ... help function.

Each time a digitizer button is pressed, the point is saved as the first point of the line (any type of line), as the lower left point of the rectangle, or as the starting text position.

The program has a build-in function that saves all of the points in an array. This function allows the user an error margin of 2 mm when he is trying to digitize a past point. The function will automatically set the new point to the former point's position. This will allow for a more accurate reproduction of the image.

A Sample Run.

```
$ GO [IAN.FALL86.GRAPHS]
$ RUN DIGITIZE
Type in the name of your output data file
Press any digitizer button to input lover left image limit
Press any digitizer button to input upper right image limit
(The boundaries of image are nov defined)
Press the digitizer button '1' at the start of a line
Press any digitizer button to input each point on line
Press the digitizer button '#' at the last point on line
(This will create a set of points to be linked by lines)
Press the digitizer button '0'
(The screen should display color change to 'blue')
Press digitizer button '5' at the start of a text line
     - the screen will wait for you to input:
                    (a) text
                    (b) text size
                    (c) text angle
(This will create text at the position digitized)
Press the digitizer button '6' at the lover left of image
Input the upper right of rectangle (any button)
(This will create a rectangle)
Press the digitizer button 'D' to display 'HELP' function
Press the digitizer button '#' to end digitizing.
```

This small exercise will create an image surrounded by a blue box with a line and text within. The user creates the line and the text. The line should be in black and the text in blue. It should also be noted here that all DIGITIZE files have the suffix ".DGT".

Program Listing.

The program DIGITIZE listing written in VAX FORTRAN 77 is as follows:

```
      PROGRAM DIGITIZE

c     Digitizer program to reproduce figures on plotter.

c     Initialize variables.

      INTEGER TYPE, COLOR, NPOINTS
      REAL SIZE,ANGLE,X,Y,XO,YO,X1,Y1,X2,Y2
      CHARACTER TEXT*40, FNAME*40, BUT*1, DUM*1, COL(4)*6
      DATA COL/' BLACK',' BLUE ',' RED  ',' GREEN'/

c     type            = type of function
c     color           = pen color
c     npoints         = no. of different points
```

```
c       size              = text size
c       angle             = text angle
c       x,y,x1,y1,x2,y2   = point positions
c       xo,yo             = origin correction values
c       text              = text
c       fname             = output file name
c       but               = button pressed
c       dum               = unused button data
c       col(4)            = four different color names

c       Get output file name to store data.

        WRITE(*,*) 'ENTER OUTPUT FILE NAME :'
        READ(*,'(A40)') FNAME
        CALL CHARCOUNT(FNAME,NC)
        OPEN (UNIT=1,FILE=FNAME(1:NC)//'.DGT',STATUS='NEW')

c       Set variables and define region to be digitized.

        XO=0.0
        YO=0.0
        COLOR=1
        NPOINTS=0
        WRITE(*,*) 'ENTER LOWER LEFT POINT :'
        READ(*,10) BUT,DUM,X1,Y1
        CALL ORIGIN(XO,YO,X1,Y1,NPOINTS)
10      FORMAT(2A1,2F6.2)
        WRITE(*,*) 'ENTER UPPER RIGHT POINT :'
        READ(*,10) BUT,DUM,X2,Y2
        CALL ORIGIN(XO,YO,X2,Y2,NPOINTS)

c       Write out screen dimensions.

        WRITE(1,20) X2-X1,Y2-Y1
20      FORMAT(1X,F6.3,1X,F6.3)

c       Set origin correction values.

        XO=X1
        YO=Y1

c       Start getting data through digitizer.

25      WRITE(*,*) 'ENTER TYPE OF LINE ("D" = HELP):'
        READ(*,10) BUT,DUM,X,Y

c       Check for button and initiate function.

        IF(BUT.EQ.'D') THEN
                CALL HELP
                GOTO 25
        ELSEIF(BUT.EQ.'0') THEN
c       Change color.
                COLOR=COLOR+1
                IF(COLOR.EQ.5) COLOR=1
                WRITE(*,*) 'NEW COLOR IS ',COL(COLOR)
                GOTO 25
        ELSEIF(BUT.EQ.'1') THEN
c       Start line.
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                WRITE(1,99) 1,COLOR,0.0,0.
                WRITE(1,20) X,Y
30              WRITE(*,*) 'ENTER LINE POINT ("#" = LAST POINT):'
```

```
                READ(*,10) BUT,DUM,X,Y
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                IF(BUT.EQ.'#') THEN
                        WRITE(1,20) X,Y
                        WRITE(1,20) -1.0,-1.0
                        GOTO 25
                ENDIF
                WRITE(1,20) X,Y
                GOTO 30
        ELSEIF(BUT.EQ.'2') THEN
c       Start vertical line.
                WRITE(*,*) 'ENTER SECOND VERTICAL POINT :'
                READ(*,10) BUT,DUM,X1,Y1
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                CALL ORIGIN(XO,YO,X1,Y1,NPOINTS)
                WRITE(1,99) 2,COLOR,0.0,0.
                WRITE(1,20) X,Y
                WRITE(1,20) X,Y1
        ELSEIF(BUT.EQ.'3') THEN
c       Start horizontal line.
                WRITE(*,*) 'ENTER SECOND HORIZONTAL POINT :'
                READ(*,10) BUT,DUM,X1,Y1
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                CALL ORIGIN(XO,YO,X1,Y1,NPOINTS)
                WRITE(1,99) 3,COLOR,0.0,0.
                WRITE(1,20) X,Y
                WRITE(1,20) X1 Y
        ELSEIF(BUT.EQ.'4') THEN
c       Start curved line.
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                WRITE(1,99) 4,COLOR,0.0,0.
                WRITE(1,20) X,Y
35              WRITE(*,*) 'ENTER CURVE POINT ("#" = LAST POINT):'
                READ(*,10) BUT,DUM,X,Y
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                IF(BUT.EQ.'#') THEN
                        WRITE(1,20) X,Y
                        WRITE(1,20) -1.0,-1.0
                        GOTO 25
                ENDIF
                WRITE(1,20) X,Y
                GOTO 35
        ELSEIF(BUT.EQ.'5') THEN
c       Enter text.
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                WRITE(*,*) 'ENTER TEXT (FROM KEYBOARD):'
                READ(*,'(A40)') TEXT
                WRITE(*,*) 'ENTER TEXT SIZE <X.XX> (FROM KEYBOARD):'
                READ(*,*) SIZE
                WRITE(*,*) 'ENTER TEXT ANGLE <XXX.> (FROM KEYBOARD):'
                READ(*,*) ANGLE
                WRITE(1,99) 5,COLOR,SIZE,ANGLE
                WRITE(1,40) X,Y,TEXT
40              FORMAT(1X,F6.3,1X,F6.3,1X,A40)
        ELSEIF(BUT.EQ.'6') THEN
c       Create rectangle.
                CALL ORIGIN(XO,YO,X,Y,NPOINTS)
                WRITE(*,*) 'ENTER UPPER RIGHT RECTANGLE POINT :'
                READ(*,10) BUT,DUM,X1,Y1
                CALL ORIGIN(XO,YO,X1,Y1,NPOINTS)
                WRITE(1,99) 1,COLOR,0.0,0.
                WRITE(1,20) X,Y
                WRITE(1,20) X,Y1
```

```
                WRITE(1,20) X1,Y1
                WRITE(1,20) X1,Y
                WRITE(1,20) X,Y
                WRITE(1,20) -1.0,-1.0
        ELSEIF(BUT.EQ.'#') THEN
c       Check for end.
                GOTO 50
        ENDIF

c       Restart loop.

        GOTO 25

c       Set close flag and close file.

50      WRITE(1,99) 0,1,0.0,0.
        WRITE(*,*) 'DIGITIZING IS COMPLETE.'
        CLOSE(UNIT=1)
99      FORMAT(1X,I1,1X,I1,1X,F4.2,1X,F4.0)

c       Terminate program.

        END

*************************************************

        SUBROUTINE ORIGIN(XO,YO,X,Y,N)

c       Subroutine to - convert to centimetres;
c                     - convert to (0,0) origin;
c                     - points saved and checked for accuracy.

        REAL X,Y,XO,YO,XX(1000),YY(1000)
        INTEGER N

c       x,y             = point to be used
c       xo,yo           = correction values
c       xx,yy           = array of all points
c       n               = no. of points

c       Convertion function.

        X=AINT(((X/10.)-XO)*100.)/100.
        Y=AINT(((Y/10.)-YO)*100.)/100.

c       Accuracy check.

        DO 10 I=1,N
                DX=(XX(I)-X)**2
                DY=(YY(I)-Y)**2
                DD=SQRT(DX+DY)
                IF(DD.LT.0.2) THEN
                        X=XX(I)
                        Y=YY(I)
                        RETURN
                ENDIF
10      CONTINUE
c       Save new points.

        N=N+1
        XX(N)=X
        YY(N)=Y
```

```
c       Return to main program.

        RETURN
        END
*****************************************

        SUBROUTINE HELP

c       Help function writing input types to screen.

        WRITE(*,*) 'TYPE       FUNCTION'
        WRITE(*,*) ' 0    =  COLOR CHANGE'
        WRITE(*,*) ' 1    =  LINE STARTING PT.'
        WRITE(*,*) ' 2    =  VERTICAL LINE PT.'
        WRITE(*,*) ' 3    =  HORIZONTAL LINE PT.'
        WRITE(*,*) ' 4    =  CURVE STARTING PT.'
        WRITE(*,*) ' 5    =  TEXT POSITION'
        WRITE(*,*) ' 6    =  LOWER LEFT RECTANGLE PT.'
        WRITE(*,*) ' #    =  QUIT'
        WRITE(*,*) ' D    =  HELP'

c       Return to main program.

        RETURN
        END
*******************************

        SUBROUTINE CHARCOUNT(TEXT,N)

        CHARACTER TEXT*40

        DO 100 J=1,40
                IF(TEXT(J:J).NE.' ') N=J
100     CONTINUE

        RETURN
        END
```

PROGRAM PLOTTER

The program function is to read data from a data file created by the DIGITIZE program and plot it on the Calcomp plotter.

Execution of Program

The program PLOTTER is executed by command:

RUN PLOTTER <CR>

in the [IAN.FALL86.GRAPHS] directory.

Once the program is started, it needs only to input the data file name of the file created by the DIGITIZE program. The name should not include the suffix ".DGT" since the program automatically appends the suffix when opening the file. The user must have an output data file and must have the Calcomp plotter set up for plotting.

How to Set up the Plotter.

The following is the procedure used to set up the Calcomp plotter for PLOTTER program:

1. Turn on plotter
2. Press ENTER
3. Tape a piece of paper to the plotter board
4. Press RELOCATE PLOT
5. Using the number pad, position further right pen onto the bottom righthand corner of the paper
6. Press ENTER
7. Press AUTO
8. Press ENTER
9. Press AUTO
10. If all is done properly, "00:00:00" will appear on screen
11. If this does not occur, turn plotter off and try again.

Program Listing.

The following is the program PLOTTER's listing, written for the Calcomp plotter:

```
      PROGRAM PLOTTER

c     Program to input data and plot onto Calcomp plotter.

c     Initialize variables.

      INTEGER TYPE,COLOR,NLEN
      REAL SIZE,ANGLE,X,Y,X1,Y1,X2,Y2,FACTX,FACTY,FACT
      CHARACTER TEXT*40,FNAME*40

c     type          = type of function
c     color         = pen color
c     nlen          = length of text
c     size          = text size
c     angle         = text angle
c     x,y,x1,y1,x2,y2 = position variables
c     text          = text
c     fname         = input file name

c     Get input file name containing plot data.

      WRITE(*,*) 'ENTER INPUT FILE NAME :'
      READ(*,'(a40)') FNAME
      CALL NCHAR(FNAME,NC)
      OPEN(UNIT=1,FILE=FNAME(1:NC)//'.DGT',STATUS='OLD')

c     Get screen dimensions and set up plotter.

10    FORMAT(1X,I1,1X,I1,1X,F4.2,1X,F4.0)
      READ(1,20) X,Y
      FACTX=1.
      FACTY=1.
C     IF(X.GT.28.5) FACTX=28./X
C     IF(Y.GT.22.) FACTY=21.5/Y
      FACT=FACTX
      IF(FACTY.LT.FACTX) FACT=FACTY
20    FORMAT(1X,F6.3,1X,F6.3)
      CALL PLOTS(0,0,6)
      IF(FACT.NE.1.) THEN
            CALL PLOT(0.,0.,-3)
      ELSE
            CALL PLOT(((28.-X)/2.),((21.5-Y)/2.),-3)
      ENDIF

c     Start recieving data to be plotted.

25    READ(1,10) TYPE,COLOR,SIZE,ANGLE

      CALL NEWPEN(COLOR)

      IF(TYPE.EQ.0) THEN
c     End of file.  Program ends.
            GOTO 50
      ELSEIF(TYPE.EQ.1) THEN
c     Plot line.
            READ(1,20) X,Y
            X=X*FACT
            Y=Y*FACT
            CALL PLOT(X,Y,3)
30          READ(1,20) X,Y
            IF(X.EQ.-1.0) GOTO 25
            X=X*FACT
            Y=Y*FACT
```

```
                CALL PLOT(X,Y,2)
                GOTO 30
        ELSEIF(TYPE.EQ.2.OR.TYPE.EQ.3) THEN
c       Plot vert. and horiz. lines.
                READ(1,20) X1,Y1
                X1=X1*FACT
                Y1=Y1*FACT
                READ(1,20) X2,Y2
                X2=X2*FACT
                Y2=Y2*FACT
                CALL PLOT(X1,Y1,3)
                CALL PLOT(X2,Y2,2)
        ELSEIF(TYPE.EQ.4) THEN
c       Plot curve.
                CALL CURVE(FACT)
        ELSEIF(TYPE.EQ.5) THEN
c       Plot text.
                READ(1,40) X,Y,TEXT
                X=X*FACT
                Y=Y*FACT
40              FORMAT(1X,F6.3,1X,F6.3,1X,A40)
                CALL NCHAR(TEXT,NLEN)
                CALL SYMBOL(X,Y,SIZE,TEXT,14,ANGLE,NLEN)
        ENDIF

c       Restart loop.

        GOTO 25

c       Close file and end plot.

50      CLOSE(UNIT=1)
        CALL PLOT(34.0,0.0,999)

c       Terminate program.

        END

*********************************

        SUBROUTINE NCHAR(TEXT,N)

c       Number of characters in text counted.

        INTEGER N,I
        CHARACTER TEXT*40

c       n       = no. of characters in text
c       i       = loop counter
c       text    = text to be used

c       Start counting non-blank characters.

        DO 10 I=1,40
                IF(TEXT(I:I).NE.' ') N=I
10      CONTINUE

c       Return to main program.

        RETURN
        END

*****************************************
```

```
      SUBROUTINE CURVE(FACT)

:     Subroutine to plot curved line onto plotter.

:     Initialize variables.
      REAL X(100),Y(100),S(1001),Q,R,TH,FACT
      REAL TX(1000),TY(1000),NQ,NR,NS(1000)
      REAL A(1000),B(1000),C(1000),D(1000)
      REAL NX(1000),NY(1000),EX(1000),WHY(1000)
      INTEGER N,E,P,K

c     Read in data and set up array.

      N=1
      K=200
10    READ(1,20) X(N),Y(N)
20    FORMAT(1X,F6.3,1X,F6.3)
      IF(X(N).EQ.-1.) THEN
              N=N-1
              X(N+1)=0.
              Y(N+1)=0.
              X(N+2)=1.
              Y(N+2)=1.
      ELSE
              X(N)=X(N)*FACT
              Y(N)=Y(N)*FACT
              N=N+1
              GOTO 10
      ENDIF

c     Process data in several routines.

      CALL GRADE(X,Y,N,S)
      DO 25 J=1,N-1
              Q=X(J)
              R=Y(J)
              E=0
              P=0
              CALL TRANSFORM(P,X,Y,J,E,Q,R,N,TH,TX,TY,NQ,NR)
              CALL ANGSLOPE(TH,S(J),S(J+1),NS(J),NS(J+1))
              CALL SPLINE(TX(1),TX(2),TY(1),TY(2),NS(J),NS(J+1),
     3                    A(J),B(J),C(J),D(J))
              CALL SEGMENT(TX,TY,J,N,K)
              NX(K+2)=X(N+1)
              NX(K+3)=X(N+2)
              NY(K+2)=Y(N+1)
              NY(K+3)=Y(N+2)
              EX(1)=TX(1)
              WHY(1)=TY(1)
              DO 35 I=1,K
                      EX(I+1)=TX(1)+(I*(TX(E+2)-TX(1)))/K
                      WHY(I+1)=A(J)+B(J)*EX(I+1)+C(J)*(EX(I+1)**2)+
     3                         D(J)*(EX(I+1)**3)
35            CONTINUE
              E=K+1
              P=1
              CALL TRANSFORM(P,EX,WHY,J,E,NQ,NR,N,-TH,NX,NY,Q,R)

:     Plot curve.

              CALL LINE(NX,NY,K+1,1,0,4)
25    CONTINUE
```

```
      Return to main program

      RETURN
      END

*****************************************
      subroutine transform(p,x,y,j,e,q,r,n,theta,tx,ty,nq,nr)

c     subroutine to transform co-ordinates when an angle of rotation
c     is to be determined by a second point or is given

c     p=point transform.le.0 (theta not given)
c     p=rotation transform, p>0 (theta given)
c     h=hypothenus
c     theta=axis rotation, in radians
c     beta=hypothenus rotation, in radians
c     gamma=total rotation, in radians
c     q=x(j)=x offset
c     nq=x-offset in terms of new origin
c     r=y(j)=y offset
c     nr=y offset in terms of new origin
c     e=# times calculation is performed (=0 if axis rotation is to be
c           calculated)
c     tx,ty=new co-ordinates
c     n=# of lines in data file

      real x(n),y(n),h,theta,beta,gamma,tx(n),ty(n),q,r,nq,nr
      integer j,p,e,n

c     calculate origin and end point of line

      if(p.le.0)then
          tx(1)=x(j)-q
          ty(1)=y(j)-r
          tx(e+2)=sqrt(((x(j+1)-x(j))**2)+((y(j+1)-y(j))**2))
          ty(e+2)=0.0
      endif

c     calculate theta

      if(p.le.0)then
          call thetaa(x(j+1),q,y(j+1),r,theta)
      endif

c     transform points other than beginning point of segment

      if(e.eq.0)then
          goto 77

c     calculation of angle between original axis and hypotenuse (beta)

      else
          do 20, i=1,e
              h=sqrt(((x(i)-q)**2)+((y(i)-r)**2))
              call betaa(x(i),q,y(i),r,beta)

c     determine the total rotation of the line segment

              call gammaa(x(i),q,beta,theta,gamma)

c     calculate the new x,y co-ordinates

              tx(i)=h*(cos(gamma))
              ty(i)=h*(sin(gamma))
```

```
20          continue

c       calculate the new q and r for use if points will later be
c       transformed back to original co-ordinates

c       if q and r = 0, then they do not need to be transformed

77          if((q.eq.0).and.(r.eq.0))then
                nq=q
                nr=r
                goto 83
            endif

c       transform q and r to the  new co-ords

            h=sqrt(((0.0-q)**2)+((0.0-r)**2))
            call  betaa(0.0,q,0.0,r,beta)
            call gammaa(0.0,q,beta,theta,gamma)
            nq=h*(cos(gamma))
            nr=h*(sin(gamma))
83      endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine thetaa(x,q,y,r,theta)

c       subroutine to calculate the angle between the original axis and the
c       new axis

c       x,y=data points
c       q,r=x and y offsets, respectively
c       theta= angle of axis rotation, in radians

        real x,q,y,r,theta

        if(((x-q).eq.0).and.((y-r).gt.0))then
            theta=1.5707963
        elseif(((x-q).eq.0).and.((y-r).lt.0))then
            theta=-1.5707963
        elseif((x-q).lt.0)then
            theta=(atan((y-r)/(x-q)))-3.1415927
        else
            theta=atan((y-r)/(x-q))
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine betaa(x,q,y,r,beta)

c       subroutine to determine the angle between the hypotenuse and the
c       x-axis

c       x,y=data points
```

```
c       q,r=x and y offsets respectively
c       beta=rotation of line segment from original axis, in radians

        real x,y,q,r,beta

        if(((x-q).eq.0).and.((y-r).gt.0))then
            beta=1.570963
        elseif(((x-q).eq.0).and.((y-r).lt.0.0))then
            beta=-1.5707963
        elseif(((x-q).eq.0).and.((y-r).eq.0))then
            beta=0.0
        else
            beta=atan((y-r)/(x-q))
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine gammaa(x,q,beta,theta,gamma)

c       subroutine to determine the total rotation of the line segment from
c       the new axis

c       x=data point
c       q=x offset
c       beta= angle of rotation of segment from original axis, in radians
c       theta=angle of rotation of new axis from original axis, in radians
c       gamma=angle of rotation of segment from new axis, in radians

        real x,q,beta,theta,gamma

        if(x.lt.q)then
            gamma=(beta-theta)-3.1415927
        else
            gamma=beta-theta
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine angslope(theta,s1,s2,ns1,ns2)

c       subroutine to change a slope to an angle, convert the angle to new
c       co-ordinates, then change it back to a slope

c       theta=axis rotation, in radians
c       s1,s2=original slopes
c       ns1,ns2=transformed slopes
c       angle1,angle2=original angle of original slopes, in radians
c       newan1,newan2=new angle of original slopes,in radians

        real theta,s1,s2,ns1,ns2,newan1,newan2,angle1,angle2

c       calculate original angles
```

```
      angle1=atan(s1)
      angle2=atan(s2)

c     calculate the new angles

      newan1=angle1-theta
      newan2=angle2-theta

c     calculate the new slopes

      ns1=tan(newan1)
      ns2=tan(newan2)
c     return to main program

77    return
      end

*******************************

      subroutine spline(x1,x2,y1,y2,s1,s2,a,b,c,d)

c     a subroutine to fit a cubic function between 2 points

      real x1,x2,y1,y2,s1,s2,a,b,c,d,num,denom

c     cubic function equations

c     y1=a+b*x1+c*(x1**2)+d*(x1**3)
c     y2=a+b*x2+c*(x2**2)+d*(x2**3)

c     slope function equations are the differential of the cubic
c     equations

c     s1=b+2*c*x1+3*d*(x1**2)
c     s2=b+2*c*x2+3*d*(x2**2)

c     mathematical manipulation produces the following equations
c     to solve for a,b,c,d

      num=(y1-y2)-(s2*(x1-x2))+(x2*(s1-s2))-(0.5*(x1+x2)*(s1-s2))

      denom=(0.5*(x2**3))-(0.5*(x1**3))+((3./2.)*(x1**2)*x2)-((1.5)
      3       *x1*(x2**2))

c     if the denominator = 0 then d is assumed to be 0

      if(denom.eq.0)then
          d=0.
      else
          d=num/denom
      endif

c     if x1=x2 then c is 0

      if((x1-x2).eq.0)then
          c=0.
      else
          c=((s1-s2)-(3*d*((x1**2)-(x2**2))))/(2*(x1-x2))
      endif

      b=s1-(2*c*x1)-(3*d*(x1**2))
```

```
      a=y1-(b*x1)-(c*(x1**2))-(d*(x1**3))

c     return to main program

      return
      end

*************************************

      subroutine myscale(array,n,min,max,length,first,fact)

c     subroutine to calculate scaling factor of an array

      real array(n+2),first,fact,max,min,delta,large,length
      integer n
c     array=array for which scaling factors are calculated
c     first=first value for axis
c     large=adjusted maximum element of array
c     fact=scaling factor
c     max=maximum array element
c     min=minimum array element
c     delta=unadjusted scaling factor
c     n=# of data points in array
c     length=axis length
c     small=adjusted first value used in fact calculation
      fact=(max-min)/length
      if(min.gt.100.0) then
        first=aint(min-fact*.5)
        max=max+fact
        fact=aint((max-first)/length)
      else
        first=aint(min)
        fact=aint((((max+5.0)-min)/length)*10.)/10.
      endif
      if(fact.eq.0.0) then
        if(min.gt.1.0) first=min-1.0
        fact=aint(4.0/length*10.)/10.
      endif
      if(min.eq.0.0.and.max.gt.100.0) fact=aint(fact*1.05)

c     return to main program

      return
      end
*************************************************************

      subroutine grade(x,y,n,s)

c     subroutine to calculate grades (slopes) at certain points by using
c     a slope parallel to a slope between the two adjacent points

c     x,y=data points
c     n=# of data lines in file
c     s=slope

      real x(n),y(n),s(n)
      integer n

c     for the slope of the first point
          if((x(2)-x(1)).eq.0)then
              s(1)=999999.999
```

```
            else
                s(1)=((y(2)-y(1))/(x(2)-x(1)))
            endif
        if(n.eq.2) goto 110
        do 100 j=2,n-1
c           if slope is infinity, assign a large value

            if((x(j+1)-x(j-1)).eq.0)then
                s(j)=999999.999
            else
                s(j)=((y(j+1)-y(j-1))/(x(j+1)-x(j-1)))
            endif
100     continue
110     continue
c           if slope is infinity, assign a large value

            if((x(n-1)-x(n)).eq.0)then
                s(n)=999999.999
            else
                s(n)=((y(n)-y(n-1))/(x(n)-x(n-1)))

            endif
c       return to main program

        return
        end

c*******************************************************************************

        subroutine angle(x,y,n,j,s)

c       subroutine to calculate the slope at each point by using angles

c       x,y=data points
c       n=# of data lines
c       s=slope at point
c       alpha=angle of slope at point
c       sigma=angle between slope angle and hypotenuse formed by point
c             and point agter
c       rho=angle formed between hypotenuse and x-axis
c       tau=angle formed between hypotenuse and y-axis
c       d=distance between two points (hypotenuse)

        real x(n),y(n),s(n),d1,d2,alpha,rho,tau,pi,mu,chi
        integer n,j

        pi=3.1414927

c       for the first and last slopes, a different calculation is made
c       based on delta y over delta x

c       for the slope at the first point

        if(j.eq.1)then
            if((x(j+1)-x(j)).eq.0)then
                s(j)=999999.999
            else
                s(j)=(y(j+1)-y(j))/(x(j+1)-x(j))
            endif
            goto 100

c       for the slope at the last point
```

```
      elseif(j.eq.n)then
          if((x(j)-x(j-1)).eq.0)then
              s(j)=999999.999
          else
              s(j)=(y(j)-y(j-1))/(x(j)-x(j-1))
          endif
          goto 100
      endif

c     calculate the distance between points

      d1=sqrt(((y(j)-y(j-1))**2)+((x(j)-x(j-1))**2))
      d2=sqrt(((y(j+1)-y(j))**2)+((x(j+1)-x(j))**2))

c     calculate the angles needed to calculate the slope

      tau=atan((x(j)-x(j-1))/(y(j)-y(j-1)))
      mu=atan((x(j+1)-x(j))/(y(j+1)-y(j)))

c     if the distance between 1 and 2, and between 2 and 3 are
c     the same, the following conditions apply

      if(d2.eq.d1)then
          if((abs(tau).eq.abs(mu)).and.(x(j-1).eq.x(j+1)))then
              s(j)=999999.999
          elseif((abs(tau).eq.abs(mu)).and.(y(j-1).eq.y(j+1)))then
              s(j)=0.00000
          else
              d2=d2+1
              write(2,*),'d2=d1'
              goto 799
          endif
          goto 100
799   endif

c     rho must have the same sign as tau for the sign of the slope to
c     work out correctly, and chi is assigned an appropiate value (tau-pi)

      if(tau.lt.0)then
          rho=-(pi/2.)-tau
          chi=-rho
      else
          rho=(pi/2.)-tau
          chi=-rho
      endif

c     calculation of alpha (angle corresponding to the slope)
c     for each case tau must be considered for four different
c     angles; 1) <pi/8
c             2) pi/8<tau<pi/4
c             3) pi/4<tau<3pi/8
c             4) tau>3pi/8

c     case1:  y1>y2<y3

1     if((y(j-1).gt.y(j)).and.(y(j+1).gt.y(j)))then
          alpha=(rho*d2)/(abs(d2+d1))

c     case2:  y1<y2>y3

2     elseif((y(j-1).lt.y(j)).and.(y(j+1).lt.y(j)))then
          alpha=(rho*d2)/(d1+d2)
```

```
c       case3a:  x1>x2>x3 and y1>y2>y3

31      elseif(((x(j-1).gt.x(j)).and.(x(j).gt.x(j+1))).and.
3           ((y(j-1).gt.y(j)).and.(y(j).gt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(chi*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(d2+d1)
            endif

c       case3b:  x1>x2>x3 and y1<y2<y3

32      elseif(((x(j-1).gt.x(j)).and.(x(j).gt.x(j+1))).and.
3           ((y(j-1).lt.y(j)).and.(y(j).lt.y(j+1))))then
            IF(ABS(TAU).LT.(PI/8.))THEN
                ALPHA=(chi*D2)/(ABS(D2-D1))
            ELSE
                alpha=(rho*d2)/(d2+d1)
            endif

c       case4:  x1<x2>x3

4       elseif((x(j-1).lt.x(j)).and.(x(j+1).lt.x(j)))then
            IF(ABS(TAU).LT.(PI/8.))THEN
                ALPHA=(rho*d2)/(abs(d2-d1))
            ELSEIF(ABS(TAU).LT.(PI/4.))THEN
                ALPHA=(chi*D2)/(ABS(D2-D1))
            ELSE
                alpha=(rho*d2)/(abs(d2+d1))
            endif

c       case5:  x1>x2<x3

5       elseif((x(j-1).gt.x(j)).and.(x(j+1).gt.x(j)))then
            if(abs(tau).lt.(pi/4.))then
                alpha=(chi*d2)/(abs(d2-d1))
            elseif(abs(tau).lt.((3./8.)*pi))then
                alpha=(rho*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(abs(d2+d1))
            endif

c       case6:  x1<x2<x3 and y1<y2<y3

6       elseif(((x(j-1).lt.x(j)).and.(x(j).lt.x(j+1))).and.
3           ((y(j-1).lt.y(j)).and.(y(j).lt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(rho*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(d2+d1)
            endif

c       case7:   x1<x2<x3 and y1>y2>y3

7       elseif(((x(j-1).lt.x(j)).and.(x(j).lt.x(j+1))).and.
3           ((y(j-1).gt.y(j)).and.(y(j).gt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(rho*d2)/(ABS(d2-d1))
            else
                alpha=(rho*d2)/(abs(d2+d1))
            endif
        endif
```

```
c       calculate slope

        s(j)=tan(alpha)

c       return to main program

100     return
        end

*******************************888

        subroutine segment(x,y,j,n,k)

c       subroutine that determines the number of segments that the spline
c       function will split the line, between two given points, into

c       x,y=data points
c       k=# of segments the line between two points is divided into
c       n=# of data lines in file

        real x(n),y(n)
        integer n,j,k

c       the number of segments depends on the length of the line

        if((abs(x(j+1)-x(j)).gt.25).or.(abs(y(j+1)-y(j)).gt.25))then
            k=50
        elseif((abs(x(j+1)-x(j)).gt.5).or.(abs(y(j+1)-y(j)).gt.5))then
            k=25
        elseif((abs(x(j+1)-x(j)).gt.1).or.(abs(y(j+1)-y(j)).gt.1))then
            k=15
        else
            k=10
        endif

c       return to main program

        return
        end

c*************************************************************************

        subroutine charcount(string,n)

c       subroutine to determine length of a character string (count ends
c       when a double blank is encountered)

        integer n
        character*(*) string

c       n=# of characters in string
c       string=character string

c       compare character values

        n=0

        do 10, i=1,len(string)
            if(string(i:i+1).gt.'  ')then
                n=n+1
            endif
```

```
10      continue
c       return to main program
        return
        end
```

CONCLUSION

The presented programs DIGITIZE and PLOTTER will provide the user with a tool to create an image of any figure or text and mechanically reproduce that image on a plotter. These programs may be used to create images of mining machinery to be included in scientific reports, to store image of plans, layouts, etc.

The program is easy to use and the presented 'sample' run is sufficient to illustrate the program capabilities even for a first time user.